КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра радіотехніки та радіоелектронних систем

«На правах рукопису»

Робота допущена до захисту в ЕК
рішенням кафедри радіотехніки та радіоелектронних систем
від ___ _____ 2024 року, протокол № ___.
Завідувач кафедри доктор фіз.-мат. наук, професор
_____ Ігор АНІСІМОВ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА
на тему:
«РОЗРОБКА ТА ВПРОВАДЖЕННЯ СИСТЕМИ ДЛЯ АВТОМАТИЗОВАНОЇ ПЕРЕВІРКИ БЕЗПЕКИ ПОСИЛАНЬ У TELEGRAM»

Виконав:
студент 4-го курсу
денної форми навчання
спеціальності 172 - Телекомунікації та радіотехніка
ОПП «Інформаційна безпека телекомунікаційних систем і мереж»
Ткаченко Олександр Олександрович _____

Науковий керівник:
к.т.н., доц. Жиров Геннадій Борисович _____

Рецензент:
д.т.н., проф. Вишнівський Віктор Вікторович _____

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____ Ткаченко Олександр

Київ – 2024

РЕФЕРАТ

Дипломна робота: 56 с., 27 рис., 5 дод., 13 джерел.

Ключові слова: TELEGRAM БОТ, БЕЗПЕКА ПОСИЛАНЬ, АВТОМАТИЗАЦІЯ, БАЗА ДАНИХ, ПЕРЕВІРКА БЕЗПЕКИ, VIRUSTOTAL, БАГАТОМОВНІСТЬ.

Об'єкт розробки - Telegram бот для автоматизованої перевірки безпеки посилань у чатах та приватних повідомленнях.

Мета роботи - розробка та впровадження Telegram бота для забезпечення безпеки користувачів шляхом автоматизованої перевірки URL-адрес на предмет шкідливого вмісту з використанням API VirusTotal.

Розроблено Telegram бот, який автоматично перевіряє URL-адреси, що надсилаються у чатах та приватних повідомленнях, на предмет їх безпеки. Бот зчитує конфігураційні параметри з файлу та ініціалізує базу даних на основі JSON файлу для збереження мовних налаштувань користувачів та груп. Для перевірки URL-адрес використовується API VirusTotal. У разі виявлення небезпечних посилань, бот автоматично видаляє їх з групових чатів та сповіщає користувачів про небезпеку. Бот також надає можливість змінювати мову інтерфейсу через інтерактивне меню. Завдяки використанню бази даних на основі JSON файлу, зберігання та оновлення мовних налаштувань здійснюється легко та зручно. Розроблений бот ефективно підвищує рівень безпеки користувачів у процесі онлайн-спілкування.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API - (Application Programming Interface) інтерфейс програмування додатків

IDE - (Integrated Development Environment) інтегроване середовище розробки

CRM - (Customer Relationship Management) управління взаємовідносинами з клієнтами

JSON - (JavaScript Object Notation) формат обміну даними

Bot API - (Telegram Bot Application Programming Interface) інтерфейс програмування додатків для Telegram ботів

URL - (Uniform Resource Locator) уніфікований вказівник ресурсу

СУБД - систем управління базами даних

ВСТУП

Розробка та впровадження Telegram бота для автоматизованої перевірки безпеки посилань є актуальним завданням у сучасних умовах, коли обсяг інформації, що передається через соціальні мережі та менеджери, стрімко зростає. Основною метою цієї дипломної роботи було створення інструменту, який забезпечить користувачам надійний захист від шкідливих URL-адрес, зменшивши ризики кіберзагроз у їхньому повсякденному спілкуванні.

Telegram бот, розроблений у рамках цієї роботи, успішно виконує свої функції завдяки використанню API VirusTotal для перевірки URL-адрес на наявність шкідливого вмісту. Це забезпечує високу точність та оперативність у виявленні потенційно небезпечних посилань. Крім того, бот має можливість автоматичного видалення повідомлень з небезпечними посиланнями у групових чатах, що додатково підвищує рівень безпеки.

Завдяки використанню бази даних на основі JSON файлу, вдалося забезпечити збереження та оновлення мовних налаштувань користувачів та груп, що робить роботу бота більш зручною та гнучкою. Це дозволяє боту адаптуватися під різноманітні потреби користувачів, забезпечуючи комфортне та інтуїтивно зрозуміле користування.

Результати тестування бота показали його високу ефективність та надійність. Бот успішно ідентифікує небезпечні посилання, сповіщає користувачів про можливу загрозу та виконує необхідні дії для запобігання поширенню шкідливого вмісту. Висока точність перевірки та швидкість обробки повідомлень забезпечують користувачам високий рівень захисту та комфорту.

З часом впровадження даного Telegram бота можна розширити, додаючи нові функціональні можливості та інтеграцію з іншими платформами та сервісами. Наприклад, можливість аналізу вкладених файлів та зображень, що надсилаються у чатах, або інтеграція з іншими сервісами кібербезпеки для більш комплексного захисту.

Загалом, розробка Telegram бота для автоматизованої перевірки безпеки посилань є важливим кроком у напрямку підвищення рівня кібербезпеки

користувачів. Вона демонструє, як сучасні технології можуть бути використані для створення інструментів, що забезпечують захист від кіберзагроз у реальному часі. Використання таких інструментів може значно зменшити ризики, пов’язані з шкідливими посиланнями, та забезпечити безпечне середовище для спілкування та обміну інформацією.

У подальшому, удосконалення алгоритмів обробки та перевірки URL-адрес, розширення функціоналу бота та інтеграція з додатковими сервісами забезпечать ще вищий рівень захисту та зроблять бота ще більш корисним та ефективним інструментом у боротьбі з кіберзагрозами.

1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО TELEGRAM БОТІВ

Telegram боти – автоматизовані програми, що взаємодіють з користувачами через месенджер Telegram. Вони виконують завдання, такі як надання інформації, обробка запитів, автоматизація бізнес-процесів та інтеграція з іншими сервісами, надаючи додаткові функції та можливості. Поява Bot API у 2015 році дозволила розробникам створювати боти, які швидко набули популярності.

Боти можуть відповідати на запити, надавати новини, керувати замовленнями, організовувати голосування та виконувати інші функції. Розробка ботів приваблює розробників завдяки простому API та наявності численних бібліотек і фреймворків.

Telegram боти використовують API для надсилання та отримання повідомлень. Основні компоненти бота – webhook або long polling, обробка команд та взаємодія з API Telegram. Вони можуть надсилати повідомлення, фото, відео, документи та створювати інтерактивні клавіатури для покращення взаємодії з користувачами.

Боти широко застосовуються в бізнесі для автоматизації процесів, підтримки клієнтів та маркетингових кампаній, знижуючи витрати та покращуючи сервіс. У новинах та медіа боти надають актуальні новини та сповіщення. В освіті вони проводять тести, надсилають навчальні матеріали та організовують опитування, спрощуючи навчання. Розважальні боти забезпечують користувачам інтерактивний досвід та цікаве дозвілля.

Таким чином, Telegram боти є потужним інструментом для автоматизації, покращення взаємодії з користувачами та надання нових можливостей для бізнесу та особистого використання. Їх розробка стає дедалі популярнішою завдяки розвитку технологій та API.

1.1. Огляд Telegram ботів та їх застосування

Telegram боти є інтегральною частиною платформи Telegram, що дозволяє автоматизувати багато завдань та забезпечувати зручність і ефективність у використанні. Вони виконують різні функції, від відповіді на прості запити до складних інтеграцій з іншими системами. З моменту запуску Bot API у 2015 році, Telegram боти набули значної популярності серед розробників і користувачів завдяки своїй простоті та широкому спектру можливостей.

Telegram боти знаходять застосування у багатьох сферах життя. Вони допомагають у бізнесі, медіа, освіті, розвагах та особистій продуктивності. Кожна з цих сфер має свої специфічні завдання, які можна успішно автоматизувати за допомогою ботів.

В бізнесі Telegram боти використовуються для автоматизації процесів і покращення обслуговування клієнтів. Вони здатні відповідати на запити клієнтів, надавати інформацію про продукти та послуги, обробляти замовлення та навіть проводити маркетингові кампанії. Наприклад, бот може автоматично відповідати на типові запитання клієнтів, знижуючи навантаження на службу підтримки. Крім того, боти можуть надсилати сповіщення про акції та знижки, що сприяє підвищенню продажів. Таким чином, боти допомагають бізнесу ефективно взаємодіяти з клієнтами та збільшувати прибуток.

В медіа-секторі Telegram боти використовуються для розповсюдження новин та оновлень. Новинні агенції та медіа-компанії активно застосовують ботів для автоматичної доставки новин користувачам. Це дозволяє отримувати найсвіжіші новини у зручний для користувачів спосіб. Наприклад, новинний бот може надсилати заголовки новин щогодини або сповіщати про важливі події у режимі реального часу. Завдяки цьому, користувачі завжди залишаються в курсі останніх подій.

Освіта є ще однією сферою, де Telegram боти стали важливим інструментом для автоматизації навчальних процесів. Вони можуть допомагати в організації дистанційного навчання, надаючи доступ до навчальних матеріалів, проводячи тести та опитування. Викладачі можуть використовувати ботів для розсилки

завдань та матеріалів, а студенти — для надсилання відповідей та отримання результатів тестів через ботів. Це значно спрощує комунікацію між викладачами та студентами, підвищуючи ефективність навчального процесу.

Розважальні Telegram боти забезпечують користувачам можливість цікаво провести час. Ігрові боти, вікторини та розважальні програми стають все більш популярними серед користувачів Telegram. Вони можуть організовувати інтерактивні ігри, конкурси та змагання, що сприяє підвищенню залученості користувачів. Наприклад, бот-вікторина може проводити щоденні ігри з питаннями на різні теми, нагороджуючи переможців віртуальними призами. Це не лише розважає користувачів, але й створює додатковий стимул для їх активності.

Особисті помічники, створені на базі Telegram ботів, допомагають користувачам у повсякденному житті. Вони можуть нагадувати про важливі події, допомагати з плануванням розкладу, надавати поради та рекомендації. Наприклад, бот-помічник може нагадати про заплановану зустріч, запропонувати список завдань на день або навіть допомогти з пошуком інформації в Інтернеті. Завдяки таким ботам, користувачі можуть краще організовувати свій час і ефективніше виконувати завдання.

Одним з важливих аспектів використання Telegram ботів є їх здатність інтегруватися з іншими сервісами та системами. Наприклад, бот може бути підключений до CRM-системи, щоб автоматично оновлювати інформацію про клієнтів, або до сервісу електронної пошти для надсилання сповіщень. Також боти можуть працювати з платіжними системами, дозволяючи користувачам здійснювати покупки та платежі безпосередньо через Telegram. Це розширює можливості ботів і робить їх незамінним інструментом у багатьох галузях.

Загалом, Telegram боти є потужним інструментом для автоматизації та покращення різних процесів. Вони надають можливість легко взаємодіяти з користувачами, забезпечуючи зручність і доступність сервісів. Завдяки постійному розвитку технологій, можливості Telegram ботів продовжують розширюватися, відкриваючи нові горизонти для їх застосування.

1.2. Інструменти та технології для розробки Telegram ботів

Розробка Telegram ботів вимагає використання певних інструментів та технологій, які забезпечують ефективність та функціональність ботів. Цей процес включає вибір мови програмування, використання відповідних бібліотек, інтеграцію з API Telegram, а також налагодження та тестування коду.

Для розробки Telegram ботів можна використовувати різні мови програмування, кожна з яких має свої переваги та недоліки. Серед найпопулярніших мов для створення ботів варто виділити:

-Python: Python є однією з найпопулярніших мов програмування для створення Telegram ботів. Він відомий своєю простотою, читабельністю та великою кількістю бібліотек, що полегшують розробку. Python також має велику спільноту розробників, що забезпечує доступність ресурсів і підтримку.

-JavaScript: JavaScript часто використовується для розробки ботів завдяки своїй популярності у веб-розробці. Бібліотека Telegraf.js є одним з найпопулярніших інструментів для створення Telegram ботів на JavaScript.

-PHP: PHP є ще однією популярною мовою для створення ботів, особливо серед веб-розробників. Бібліотека MadelineProto дозволяє легко створювати та керувати Telegram ботами на PHP.

-Java: Java є потужною мовою програмування, яка також використовується для створення ботів. Бібліотека TelegramBots надає всі необхідні інструменти для роботи з API Telegram.

-C#: Мова програмування C# також використовується для створення ботів, зокрема завдяки бібліотеці Telegram.Bot, яка надає широкий функціонал для роботи з Telegram API.

Для цього проекту було обрано мову програмування Python [1], оскільки вона має декілька суттєвих переваг у порівнянні з іншими мовами. По-перше, Python є дуже простою та зрозумілою мовою, що дозволяє швидко розпочати розробку та легко підтримувати код. По-друге, Python має багато бібліотек та фреймворків, що значно спрощують процес розробки. Однією з таких бібліотек є TeleBot

(PyTelegramBotAPI) [2], яка забезпечує зручний інтерфейс для взаємодії з Telegram API.

Python також має велику спільноту розробників [3], що забезпечує доступ до численних ресурсів, прикладів коду та підтримки. Це робить Python ідеальним вибором для створення Telegram ботів, особливо для розробників-початківців.

Для розробки бота використовувалося інтегроване середовище розробки (IDE) PyCharm Community Edition [4], яке наведено на рис.1.1. PyCharm є одним з найпопулярніших середовищ розробки для Python завдяки своїй зручності та потужному функціоналу. Він підтримує автодоповнення коду, рефакторинг, інтеграцію з системами контролю версій, вбудовані інструменти для тестування та налагодження, що робить його найзручнішою програмою для розробки Telegram ботів. PyCharm також має багато плагінів, які можуть розширити його функціонал [13] і зробити процес розробки ще більш ефективним.

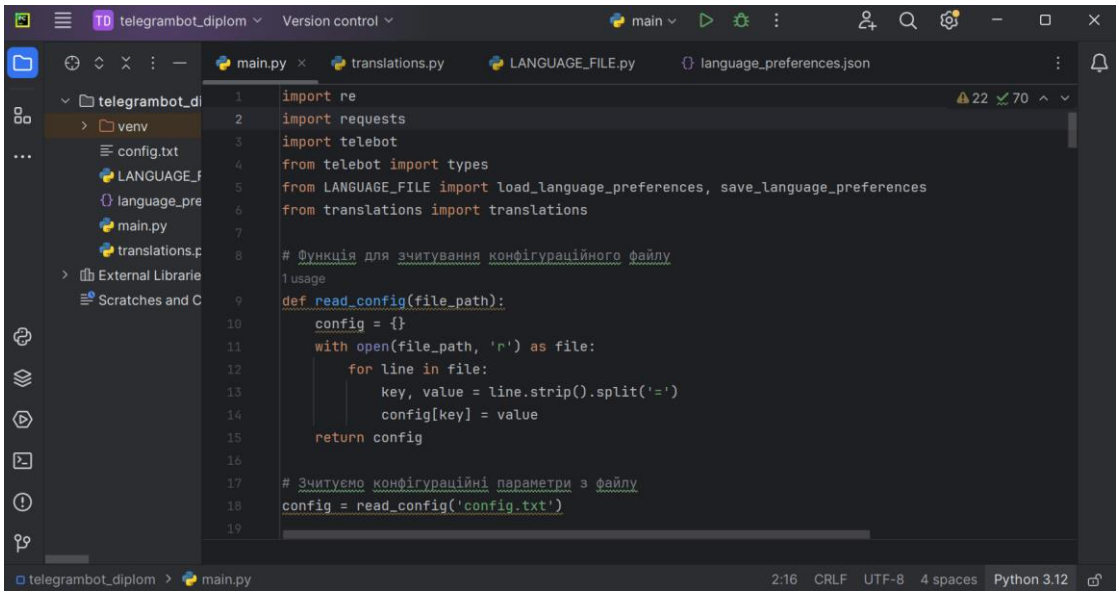

Рисунок 1.1 – Приклад вигляду програми PyCharm

Telegram надає розробникам відкритий API, який дозволяє створювати ботів та взаємодіяти з платформою. API підтримує різні методи для відправки повідомлень, обробки команд, управління чатами та інших функцій. Щоб розпочати роботу з Telegram API [6], необхідно створити бота через BotFather —

офіційного бота Telegram для створення нових ботів та отримання токену доступу. Приклад отримання ключу доступу наведений на рис.1.2.

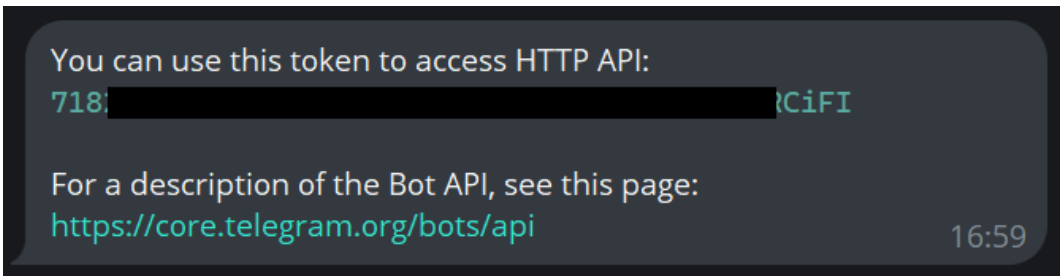

Рисунок 1.2 – Унікальний API ключ доступу до HTTPS Telegram

Telegram API дозволяє ботам надсилати текстові повідомлення, фото, відео, документи, створювати опитування, інтерактивні кнопки та багато іншого. Він також підтримує Webhook та long polling — два основних методи отримання оновлень від серверу Telegram.

Для зберігання та передачі даних у Telegram ботах часто використовується формат JSON [8]. Він є легким для читання людиною та зручним для використання у програмах. JSON використовується для зберігання налаштувань бота, таких як мовні переваги користувачів, дані про сеанси та іншу інформацію. Завдяки своїй простоті, JSON став стандартом де-факто для обміну даними у веб-додатках.

Однією з потужних можливостей Telegram ботів є інтеграція з іншими сервісами через API [11]. Наприклад, для забезпечення безпеки посилань бот може використовувати API VirusTotal, який можна побачити на рис.1.3, для перевірки URL-адрес на наявність шкідливих елементів. Інтеграція з платіжними системами дозволяє ботам обробляти платежі [10], а з CRM-системами — оновлювати інформацію про клієнтів.

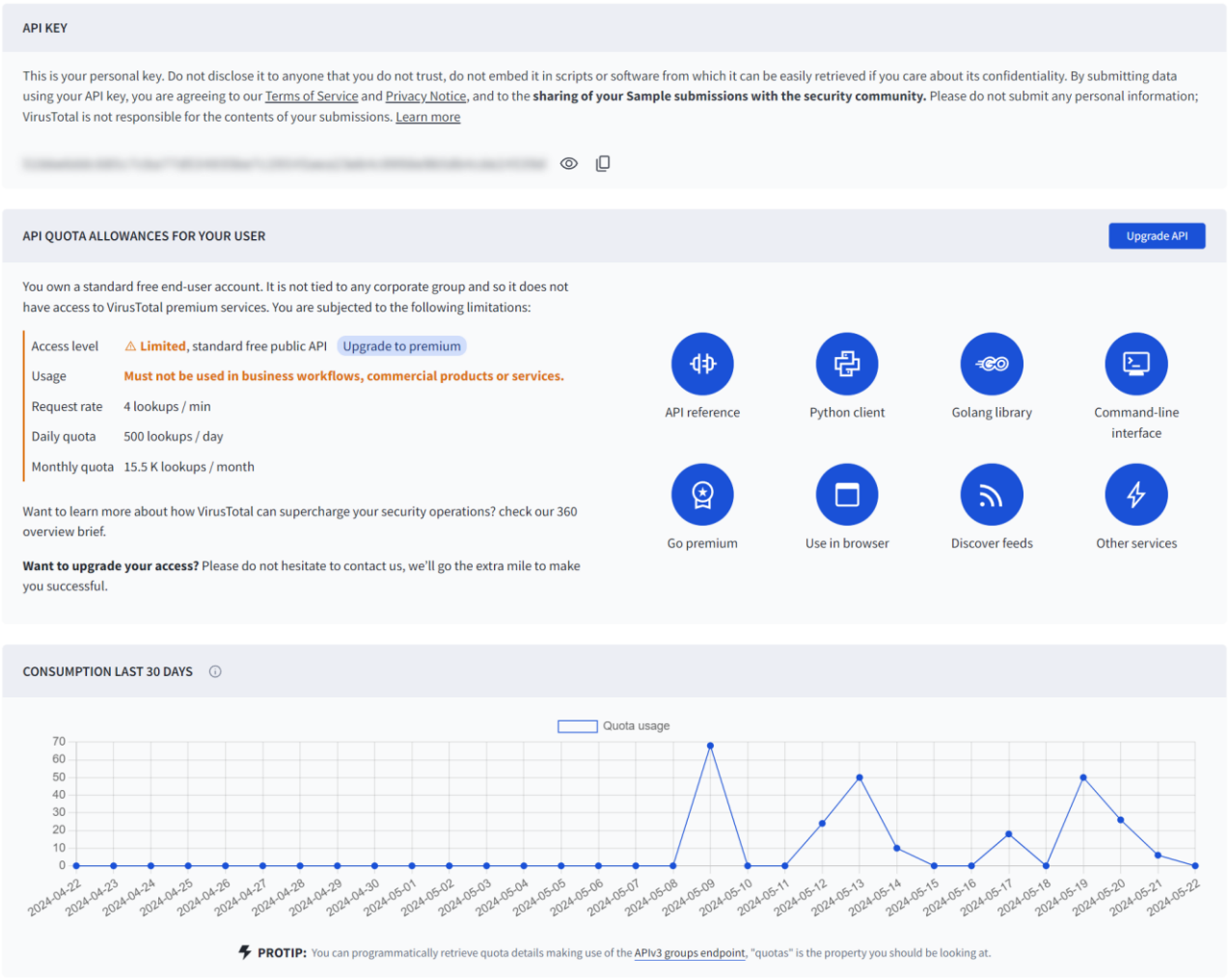

Рисунок 1.3 – Вигляд офіційної сторінки VirusTotal API ключа. Також детальна інформація про рівень доступу, обмеження на кількість запитів, а також графік використання квоти за останні 30 днів.

Інші популярні сервіси, з якими можуть інтегруватися Telegram боти, включають Google Sheets для роботи з таблицями та різні сервіси для надсилання сповіщень, такі як Twilio або SendGrid.

Налагодження та тестування є важливими етапами розробки Telegram ботів. Вони забезпечують правильність роботи бота та виявлення помилок. Для цього можуть використовуватися різні інструменти, такі як логування, відладчики та автоматизовані тести. Наприклад, логування дозволяє відслідковувати роботу бота та виявляти проблеми у реальному часі, тоді як відладчики допомагають знаходити та виправляти помилки у коді.

Таким чином, розробка Telegram ботів включає використання різноманітних інструментів та технологій [9], кожен з яких відіграє важливу роль у створенні функціонального та надійного бота. Від вибору мови програмування та бібліотек до інтеграції з іншими сервісами та налагодження, кожен етап є важливим для успіху проекту.

2. ОПИС РОЗРОБКИ TELEGRAM БОТА ДЛЯ ПЕРЕВІРКИ БЕЗПЕКИ ПОСИЛАНЬ

Цей розділ описує процес розробки Telegram бота для перевірки безпеки посилань у чатах. Метою проекту є створення інструменту, який автоматично аналізує URL-адреси за допомогою сервісу VirusTotal. Проект реалізовано на Python з використанням середовища розробки PyCharm Community Edition.

Python було обрано через простоту, читабельність та численні бібліотеки, що полегшують розробку. Основною бібліотекою є TeleBot (PyTelegramBotAPI), яка надає зручний інтерфейс для роботи з Telegram API. Python забезпечує швидкий старт розробки та легке підтримання коду завдяки великій спільноті розробників.

PyCharm Community Edition пропонує інструменти для автодоповнення коду, налагодження, інтеграції з системами контролю версій, рефакторингу та багато інших функцій, що роблять його зручним для розробки Telegram ботів.

Бот виконує такі основні функції: отримання повідомлень з посиланнями, перевірка безпеки посилань через API VirusTotal та надсилання результатів користувачам. Він автоматично сканує повідомлення, перевіряє URL на наявність шкідливого контенту і повідомляє користувачів про результати.

Відсутність такого бота може призвести до значних ризиків для користувачів чатів. Без автоматичної перевірки посилань існує висока ймовірність того, що користувачі можуть натрапити на шкідливі ресурси, що можуть містити віруси, фішингові сайти чи інший небезпечний контент. Це може призвести до втрати особистих даних, компрометації облікових записів та інших негативних наслідків.

Проект включає Telegram Bot API для обміну повідомленнями, VirusTotal API для перевірки посилань та бібліотеку TeleBot для взаємодії з Telegram Bot API. Розробка включала налаштування середовища, створення бота через BotFather, інтеграцію з VirusTotal API та написання коду для основних функцій бота.

Розробка бота показала ефективність використання Python та PyCharm для створення функціональних рішень. Бот надає можливість автоматично перевіряти посилання, забезпечуючи захист від шкідливого контенту. Використання сучасних

інструментів дозволило швидко реалізувати проект та відкриває нові можливості для його розвитку.

2.1. Архітектура та функціональні можливості бота

Розробка Telegram бота для перевірки безпеки посилань передбачає створення надійної та зручної структури, яка дозволяє ефективно обробляти та аналізувати URL-адреси, надані користувачами. Основна мета цього бота – забезпечити безпеку користувачів шляхом автоматичної перевірки посилань на наявність шкідливого контенту за допомогою сервісу VirusTotal.

Архітектура бота базується на використанні мови програмування Python, бібліотеки TeleBot для роботи з Telegram API та сервісу VirusTotal для перевірки посилань. Основні компоненти архітектури включають:

-Telegram Bot API: Інтерфейс, через який бот взаємодіє з платформою Telegram, дозволяючи отримувати та надсилати повідомлення, обробляти команди користувачів і керувати чатами.

-VirusTotal API [5]: Сервіс для перевірки URL-адрес на наявність шкідливих елементів. Бот надсилає запити до цього сервісу і отримує результати аналізу, які використовуються для інформування користувачів про безпеку посилань.

-Конфігураційні файли: Використовуються для зберігання налаштувань бота, таких як токен доступу до Telegram API та ключ API для сервісу VirusTotal.

-TeleBot (PyTelegramBotAPI) [7]: Основна бібліотека, що забезпечує простий інтерфейс для інтеграції з Telegram API, обробки повідомлень і реалізації функціональності бота.

-Обробники повідомлень і команд: Код, що відповідає за обробку вхідних повідомлень та команд користувачів, перевірку посилань і надання результатів.

Telegram бот для перевірки безпеки посилань автоматично перевіряє всі повідомлення, що містять URL-адреси [12] код який це робить зображено на рис. 2.1, використовуючи регулярні вирази для їх виявлення.

```python
@bot.message_handler(regexp=r"https?://[^\s]+")
def handle_links(message):
    urls = re.findall(r"https?://[^\s]+", message.text)
    for url in urls:
        if check_url_vt(url):
            bot.reply_to(message, "Це посилання небезпечне.")
        else:
            bot.reply_to(message, "Це посилання безпечне.")
```

Рисунок 2.1 – Спрощений вигляд коду, який відповідає за виявлення посилань в повідомленнях

Використовуючи VirusTotal API, бот перевіряє кожну URL-адресу, надсилаючи запити до сервісу та отримуючи детальні результати аналізу, цей процес відображено на рис. 2.2.

```python
# Перевірка URL через VirusTotal
2 usages
def check_url_vt(url):
    headers = {"x-apikey": VT_API_KEY}
    params = {"url": url}
    response = requests.post( url: "https://www.virustotal.com/api/v3/urls", headers=headers, data=params)
    if response.status_code == 200:
        json_response = response.json()
        analysis_id = json_response["data"]["id"]
        analysis_url = f"https://www.virustotal.com/api/v3/analyses/{analysis_id}"
        analysis_response = requests.get(analysis_url, headers=headers)
        if analysis_response.status_code == 200:
            analysis_results = analysis_response.json()
            stats = analysis_results["data"]["attributes"]["stats"]
            if stats["malicious"] > 0:
                return True
    return False
```

Рисунок 2.2 – Частина коду яка перевіряє посилання

Бот інформує користувачів про те, чи є посилання безпечним чи небезпечним, за допомогою повідомлень у чаті. Якщо посилання виявляється шкідливим, бот може видалити відповідне повідомлення і надіслати попередження.

Відсутність такого бота може значно знизити рівень безпеки в чатах, особливо в групах з великою кількістю учасників. Це може призвести до поширення шкідливого програмного забезпечення, яке може вразити пристрої

користувачів, зокрема викрадення особистих даних, паролів, або інших конфіденційних інформацій.

Бот підтримує багатомовний інтерфейс, дозволяючи користувачам вибирати мову взаємодії. Мовні налаштування зберігаютьсяу конфігураційних файлах, та потім завантажуються при надсиланні команд, що відображено на рис. 2.3 і можуть бути змінені.

```
26  # Функція для отримання перекладу
    16 usages
27  def get_translation(lang, key):
28      return translations.get(lang, translations[default_language]).get(key, key)
29
30  # Завантаження мовних налаштувань користувачів і груп
31  language_preferences = load_language_preferences()
32  user_languages = language_preferences.get("user_languages", {})
33  group_languages = language_preferences.get("group_languages", {})
```

Рисунок 2.3 – Частина коду яка відповідає за завантаження мовного вибору користувача

Бот може виконувати адміністративні дії, такі як видалення небезпечних повідомлень у групових чатах, надсилання сповіщень адміністраторам і управління налаштуваннями груп.

2.2. Використання API VirusTotal для перевірки URL-адрес

Щоб забезпечити безпеку користувачів у чатах, Telegram бот був інтегрований з API VirusTotal, що дозволяє автоматично перевіряти URL-адреси на наявність шкідливого контенту. VirusTotal є популярним онлайн-сервісом, який збирає дані з численних антивірусних програм та інших інструментів для забезпечення комплексного аналізу веб-ресурсів.

Інтеграція з API VirusTotal дає змогу боту надсилати запити для перевірки URL-адрес і отримувати результати аналізу. Для цього потрібно виконати кілька кроків. Спершу слід зареєструватися на сайті VirusTotal та отримати ключ API, необхідний для автентифікації запитів до сервісу. Ключ API зберігається, та зчитується, що можна побачити на рис. 2.4, у конфігураційному файлі разом з ключем API Telegram.

```python
# Зчитуємо конфігураційні параметри з файлу
config = read_config('config.txt')

# Ініціалізуємо бота з токеном з конфігураційного файлу
bot = telebot.TeleBot(config['TELEGRAM_BOT_TOKEN'])
VT_API_KEY = config['VIRUSTOTAL_API_KEY']  # Ключ для доступу до API VirusTotal
```

Рисунок 2.4 – Зчитування Telegram та VirusTotal API ключів

Наступним кроком є створення функції для перевірки URL. Функція check_url_vt відповідає за надсилання запиту до VirusTotal та обробку відповіді. Цей процес відображено на рис. 2.5. Вона надсилає URL-адресу до сервісу, отримує ідентифікатор аналізу, а потім запитує результати аналізу.

```python
# Перевірка URL через VirusTotal
2 usages
def check_url_vt(url):
    headers = {"x-apikey": VT_API_KEY}
    params = {"url": url}
    response = requests.post( url: "https://www.virustotal.com/api/v3/urls", headers=headers, data=params)
    if response.status_code == 200:
        json_response = response.json()
        analysis_id = json_response["data"]["id"]
        analysis_url = f"https://www.virustotal.com/api/v3/analyses/{analysis_id}"
        analysis_response = requests.get(analysis_url, headers=headers)
        if analysis_response.status_code == 200:
            analysis_results = analysis_response.json()
            stats = analysis_results["data"]["attributes"]["stats"]
            if stats["malicious"] > 0:
                return True
    return False
```

Рисунок 2.5 – Перевірка посилань на безпеку

Функція перевірки URL, яку зображено на рис 2.6, інтегрується в обробник повідомлень бота. Коли бот отримує повідомлення з посиланням, він викликає функцію check_url_vt для перевірки безпеки URL та відповідно реагує на результати аналізу.

```
@bot.message_handler(regexp=r"https?://[^\s]+")
def handle_links(message):
    urls = re.findall(r"https?://[^\s]+", message.text)
    for url in urls:
        if check_url_vt(url):
            bot.reply_to(message, "Це посилання небезпечне.")
        else:
            bot.reply_to(message, "Це посилання безпечне.")
```

Рисунок 2.6 – Скорочений вигляд функції яка відповідає за обробку повідомлень з посиланнями

Використання API VirusTotal надає боту кілька суттєвих переваг. VirusTotal агрегує результати з багатьох антивірусних програм та інструментів безпеки, що дозволяє отримати комплексну оцінку безпеки URL-адреси. Інтеграція з API дозволяє автоматично перевіряти всі посилання, що надсилаються в чатах, без необхідності ручного втручання. Завдяки швидкій обробці запитів, бот може миттєво реагувати на потенційні загрози, повідомляючи користувачів про небезпечні посилання. Використання перевіреного сервісу для аналізу URL-адрес забезпечує високу точність результатів та захист від помилкових спрацьовувань.

Інтеграція API VirusTotal з Telegram ботом для перевірки посилань забезпечує ефективний захист користувачів від шкідливих веб-ресурсів. Використання сучасних технологій дозволяє автоматизувати процес перевірки та забезпечити високу надійність і точність результатів, роблячи чат безпечнішим місцем для спілкування.

2.3. Збереження мовних налаштувань користувачів у базі даних

Для забезпечення зручності користувачів і покращення функціональності Telegram бота, було реалізовано збереження мовних налаштувань користувачів у базі даних. Це дозволяє боту надавати відповіді та повідомлення на тій мові, яку обрав користувач або адміністратор групи.

Для збереження мовних налаштувань користувачів було обрано JSON файл як простий та ефективний спосіб зберігання даних. Використання JSON забезпечує

легкість читання і запису даних, а також простоту у реалізації без необхідності налаштування складних систем управління базами даних (СУБД).

Початкові налаштування зберігаються у файлі language_preferences.json, який містить інформацію про мовні налаштування як для окремих користувачів, так і для груп.

При запуску бота перевіряється наявність файлу language_preferences.json. Якщо файл не існує, створюється новий файл з початковими значеннями, цей процес відображено на рис. 2.7.

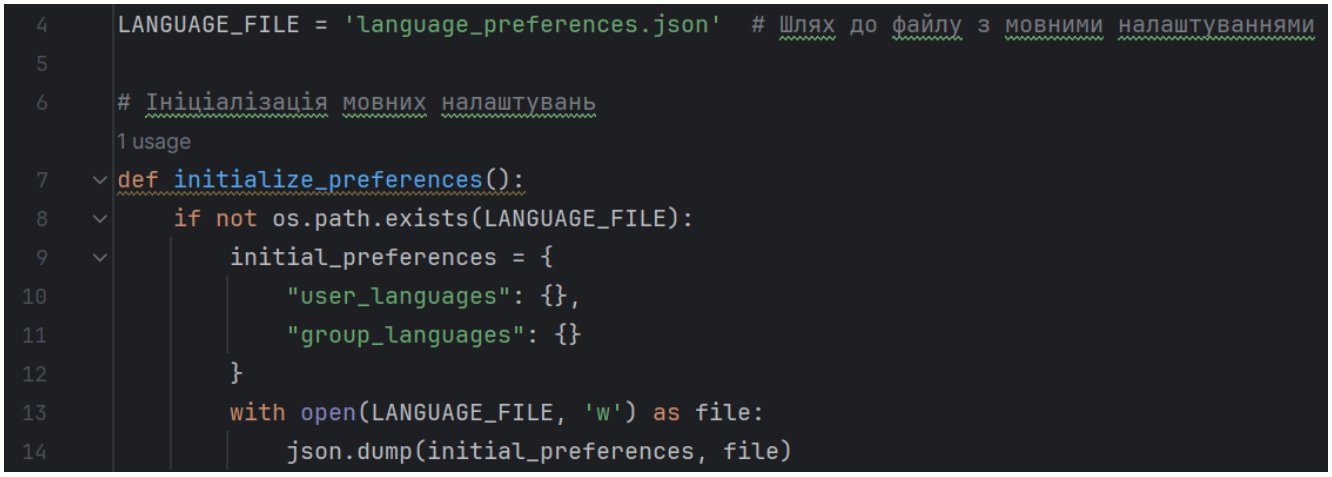

Рисунок 2.7 – Частина коду на якій показано як знаходиться шлях до файлу з мовними налаштуваннями та їх ініціалізація

Для завантаження та збереження мовних налаштувань використовуються функції load_language_preferences та save_language_preferences. Ці функції забезпечують зчитування даних з файлу та збереження оновлених налаштувань. Це можна побачити на рис. 2.8.

```python
# Завантаження мовних налаштувань з файлу
2 usages
def load_language_preferences():
    if os.path.exists(LANGUAGE_FILE):
        with open(LANGUAGE_FILE, 'r') as file:
            return json.load(file)
    return {"user_languages": {}, "group_languages": {}}

# Збереження мовних налаштувань до файлу
3 usages
def save_language_preferences(prefs):
    with open(LANGUAGE_FILE, 'w') as file:
        json.dump(prefs, file, indent=4, ensure_ascii=False)
```

Рисунок 2.8 – Завантаження та збереження мовних налаштувань в файлі-базі даних language_preferences.json

Для встановлення та завантаження мовних налаштувань користувачів та груп використовуються функції set_user_language та set_group_language, відповідно, це можна побачити на рис. 2.9. Ці функції оновлюють налаштування у файлі language_preferences.json.

```python
# Встановлення мови для користувача
1 usage
def set_user_language(user_id, lang):
    user_languages[str(user_id)] = lang
    save_language_preferences({"user_languages": user_languages, "group_languages": group_languages})

# Встановлення мови для групи
1 usage
def set_group_language(chat_id, lang):
    group_languages[str(chat_id)] = lang
    save_language_preferences({"user_languages": user_languages, "group_languages": group_languages})

# Отримання мови користувача
6 usages
def get_user_language(user_id):
    return user_languages.get(str(user_id), default_language)

# Отримання мови групи
7 usages
def get_group_language(chat_id):
    return group_languages.get(str(chat_id), default_language)
```

Рисунок 2.8 – Встановлення та завантаження мовних налаштувань користувачів

Бот використовує збережені мовні налаштування для надання відповідей та повідомлень користувачам. Коли користувач надсилає команду або повідомлення, бот звертається до файлу налаштувань, щоб визначити, якою мовою потрібно відповісти. Приклад як виглядає завантаження вибраної мови користувачем, на команді /help можна побачити на рис. 2.9.

```
104  # Обробник команди /help
105  @bot.message_handler(commands=["help"])
106  def help_command(message):
107      if message.chat.type == "private":
108          user_id = message.from_user.id
109          lang = get_user_language(user_id)
110      else:
111          chat_id = message.chat.id
112          lang = get_group_language(chat_id)
113      bot.send_message(
114          message.chat.id,
115          get_translation(lang, key: "help")
116      )
```

Рисунок 2.9 – Завантаження обраної мови користувачем в команді /help

Збереження мовних налаштувань у файлі забезпечує гнучкість і зручність для користувачів, дозволяючи їм отримувати відповіді та повідомлення на обраній мові. Це підвищує загальний рівень користувацького досвіду та робить взаємодію з ботом більш комфортною та ефективною.

Файл language_preferences.json містить дані про мовні налаштування користувачів та груп у форматі JSON. Вмісту файлу можна побачити на рис. 2.10.

Рисунок 2.10 – Як вигладає вміст файлу-бази даних language_preferences.json

Де 587096106 ID користувача а uk вибрана ним мова. Також можна побачити приклад ID групи -1001247417889 і також вибраної мови es, тобто Іспанської.

3. РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ БОТА

Для початку було створено логотип для боту у програмі Photoshop, його можна побачити на рис.3.1.

Логотип має простий та зрозумілий дизайн. Щит символізує захист, а силует людини вказує на орієнтацію на користувачів. Використання зеленої палітри додає відчуття безпеки та надійності.

Рисунок 3.1 – Логотип який було створено в программному забезпеченні Photoshop

Далі було створено оболонку боту за допомогою Телеграм боту BotFather, що можна побачити на рис. 3.2. Цей офіційний бот генерує Telegram API ключ, та за його допомогою додається назва, лого, опис, та швидкі команди, додавання команд відображено на рис. 3.3.

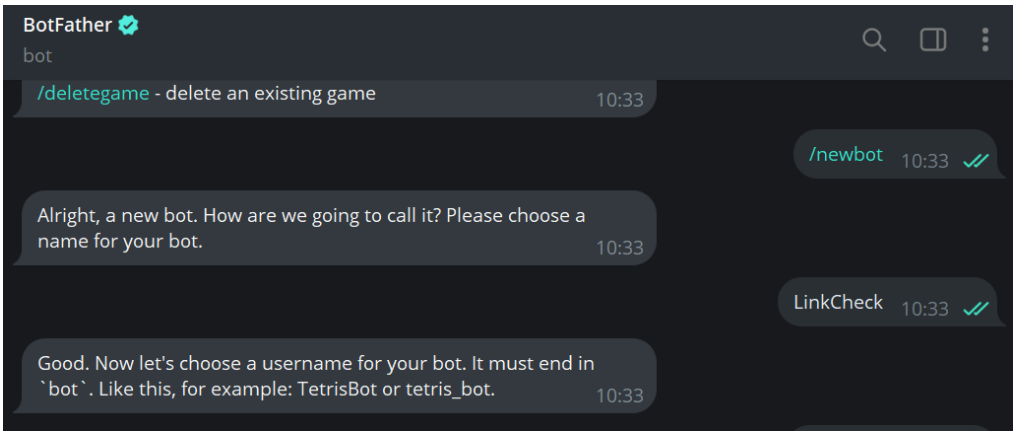

Рисунок 3.2 – Створення нового бота за допомогою BotFather. Приклад вибору імені та імені користувача для нового бота.

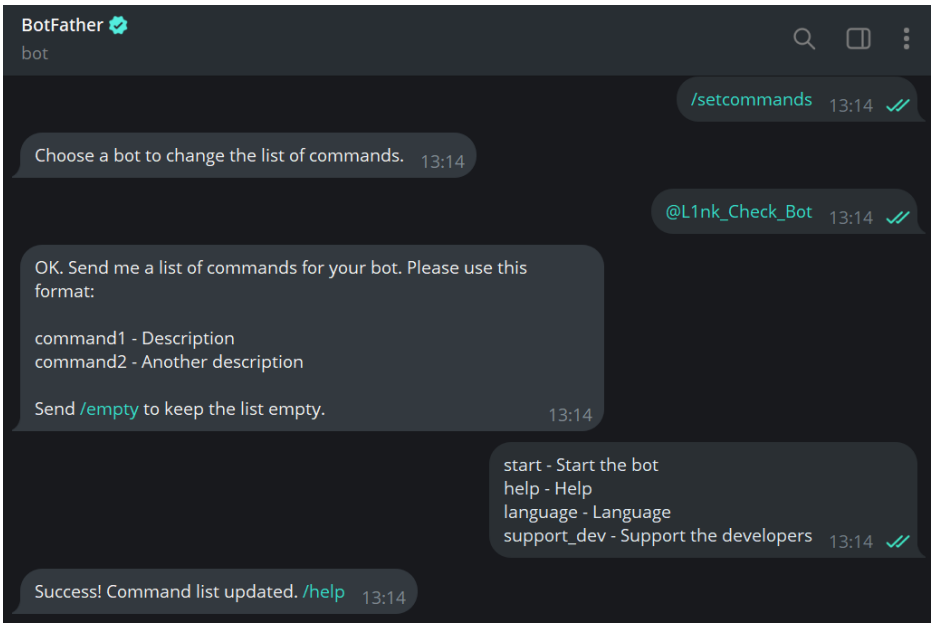

Рисунок 3.3 – Налаштування списку команд для нового бота через BotFather. Приклад додавання команд для бота LinkCheck.

Основний функціонал Telegram бота для перевірки безпеки посилань було реалізовано за допомогою мови програмування Python та бібліотеки TeleBot (PyTelegramBotAPI). Нижче наведено весь код боту, який відповідає за взаємодію бота з користувачами, обробку повідомлень, перевірку посилань та забезпечення багатомовної підтримки:

-основний код бота наведений в додатку;

-код файлу з перекладами (Додаток А). Цей файл містить переклади текстів для багатомовної підтримки бота;

-код файлу для роботи з базою даних (Додаток Б). Цей файл містить функції для роботи з JSON файлом, який зберігає мовні налаштування користувачів і груп;

-код бази даних (Додаток В). Файл language_preferences.json містить дані про мовні налаштування користувачів та груп у форматі JSON. Цей файл забезпечує збереження інформації про вибір мови для кожного користувача та групи, що дозволяє боту правильно відповідати на повідомлення, враховуючи мовні вподобання.

Загалом, реалізація функціоналу бота включає обробку команд, перевірку безпеки URL-адрес, збереження та використання мовних налаштувань

користувачів і груп, що робить бота потужним інструментом для забезпечення безпеки та зручності користувачів у Telegram чатах.

3.1. Ініціалізація та конфігурація бота

Для початку роботи Telegram бота необхідно виконати його ініціалізацію та конфігурацію. Це включає налаштування середовища розробки, отримання необхідних ключів доступу та налаштування конфігураційних файлів.

Для взаємодії з Telegram API необхідно створити нового бота за допомогою офіційного бота Telegram – BotFather. Після створення бота BotFather надає унікальний токен доступу, який використовується для автентифікації запитів до Telegram API.

Також потрібно отримати API ключ від сервісу VirusTotal для перевірки безпеки URL-адрес. Це можна зробити, зареєструвавшись на сайті VirusTotal та створивши новий ключ API у своєму акаунті.

Всі отримані ключі доступу та інші налаштування зберігаються у конфігураційному файлі config.txt. Це дозволяє легко змінювати налаштування бота без необхідності внесення змін до основного коду.

Для коду див. Додаток А

Після налаштування середовища та отримання необхідних ключів, бот ініціалізується, це видно на рис. 3.4.

```
9   def read_config(file_path):
10      config = {}
11      with open(file_path, 'r') as file:
12          for line in file:
13              key, value = line.strip().split('=')
14              config[key] = value
15      return config
```

Рисунок 3.4 – Ініціалізація та отримання необхідних ключів

В цьому коді зчитуються конфігураційні параметри з файлу config.txt, а потім ініціалізується об'єкт бота з використанням токену доступу. Бібліотека TeleBot використовується для створення бота та обробки команд користувачів.

Таким чином, процес ініціалізації та конфігурації Telegram бота включає налаштування середовища розробки, отримання ключів доступу та налаштування

конфігураційних файлів. Це дозволяє забезпечити правильну роботу бота та його взаємодію з користувачами.

3.2. Обробка команд та повідомлень

Команда /start

Команда /start ініціює взаємодію користувача з ботом. Вона виконується по-різному залежно від контексту: у приватному чаті або в груповому чаті.

У приватному чаті: Бот вітає користувача і пропонує додати його до групи або обрати мову. Показуються дві кнопки: "Додати до чату" та "Обрати мову". На рис. 3.5 результат надсилання команди старт.

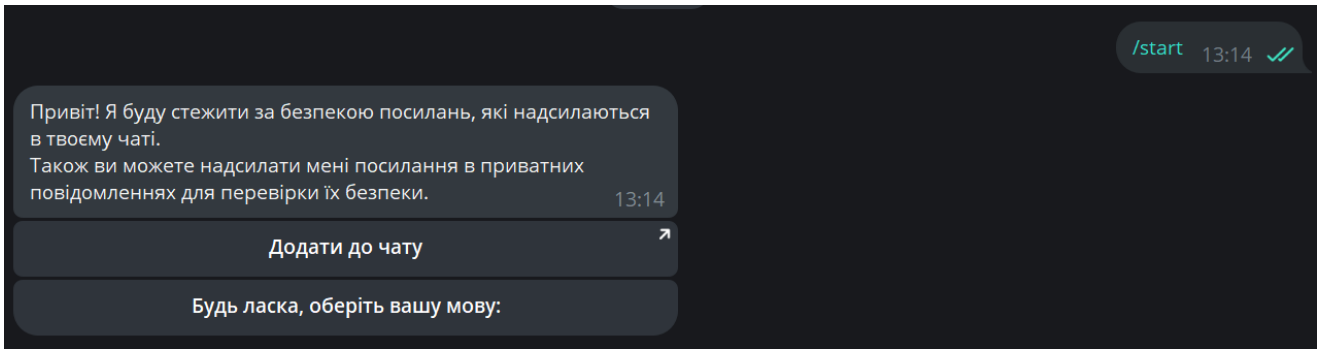

Рисунок 3.5 – Відповідь бота LinkCheck на команду /start. Привітальне повідомлення та інструкції щодо додавання бота до чату та вибору мови.

У груповому чаті: Бот дякує за додавання до групи і пропонує обрати мову. Показується кнопка "Обрати мову". На рис. 3.6 результат додавання боту в групу.

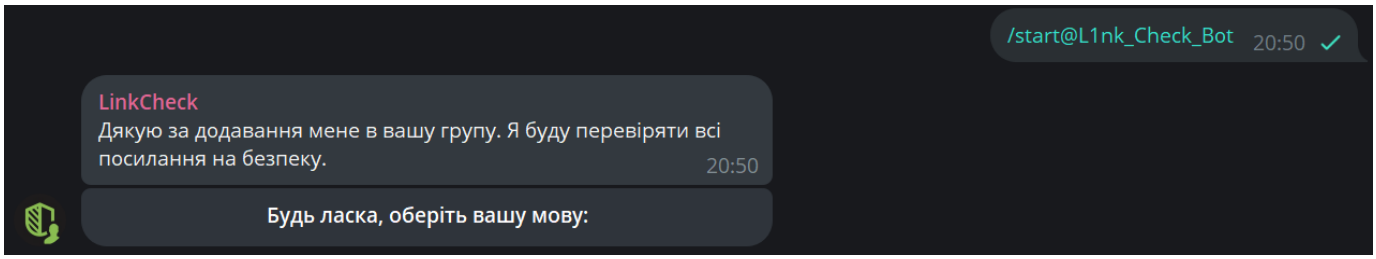

Рисунок 3.6 – Повідомлення бота LinkCheck після додавання до групи.

Бот підтверджує, що буде перевіряти всі посилання на безпеку, і запитує вибір мови.

Команда /help

Команда /help надає користувачу інформацію про те, як користуватися ботом. Виводиться список доступних команд та короткий опис їх функцій. На рис. 3.7 зображений вигляд результату команди help.

Результат цієї команди однаковий як в приватних повідомленнях так і в групі.

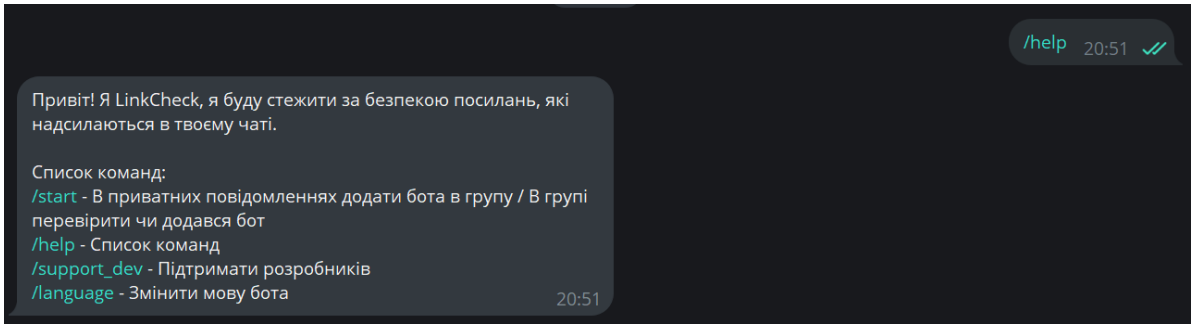

Рисунок 3.7 – Відповідь бота LinkCheck на команду /help.

Бот надає список доступних команд та їхні описи.

Команда /support_dev

Команда /support_dev призначена для підтримки розробників. Вона надає інформацію про те, як користувач може підтримати проект. Вигляд результату команди наведений на рис. 3.8.

І в приватних повідомленнях і в групі бот надсилає повідомлення з інформацією про підтримку, включаючи деталі про донати.

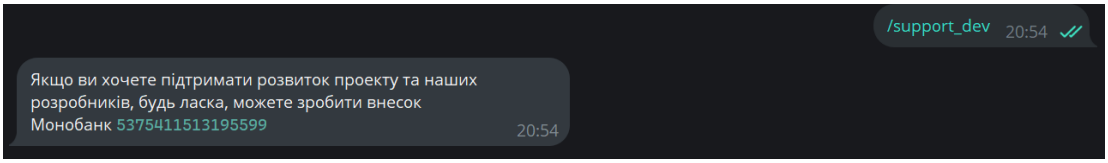

Рисунок 3.8 – Відповідь бота LinkCheck на команду /support_dev.

Інформація про те, як можна підтримати проект та розробників, з наданням реквізитів для внеску.

Команда /language

Команда /language дозволяє користувачу обрати мову для взаємодії з ботом. Показується список доступних мов, це можна побачити на рис. 3.9.

У приватному чаті: Користувач може обрати мову зі списку. Показуються кнопки з мовами: "English", "Українська", "日本語", "Español", "中文".

Рисунок 3.9 – Відповідь бота LinkCheck на команду /language. Інтерфейс вибору мови з підтримкою 5 мов. Також вигляд повідомлення, яке надсилає бот, після вибору мови.

У груповому чаті: Користувач-адміністратор може обрати мову для всієї групи зі списку мов. Якщо це намагається зробити не адміністратор, то висвітиться повідомлення що у користувача немає прав на зміну мови, це повідомлення можна побачити на рис. 3.10.

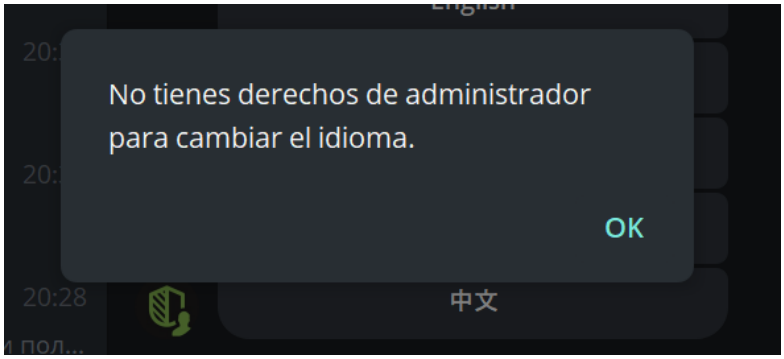

Рисунок 3.10 – Повідомлення про помилку від бота LinkCheck при спробі змінити мову без прав адміністратора в групі на іспанській мові.

Бот автоматично обробляє повідомлення, що містять URL-адреси, і перевіряє їх на безпеку за допомогою VirusTotal API.

У приватному чаті: Користувач надсилає повідомлення з посиланням, і бот відповідає, чи є посилання безпечним або небезпечним, це можна побачити на рис. 3.11.

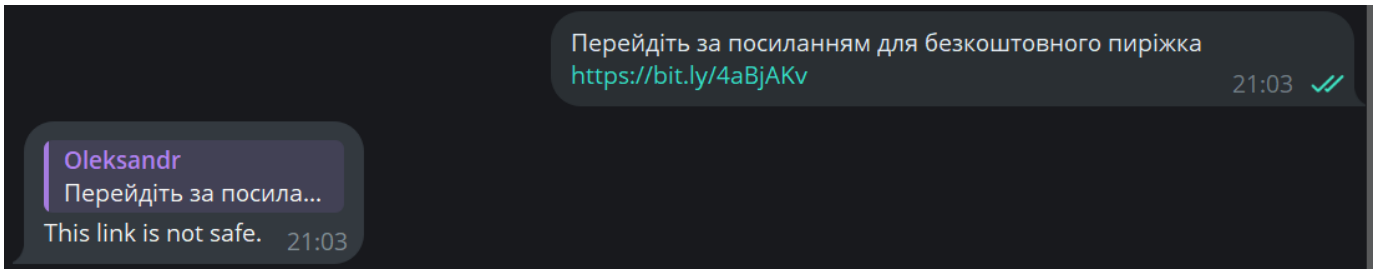

Рисунок 3.11 – Приклад роботи бота LinkCheck в приватних повідомленнях.

Бот виявляє небезпечне посилання та попереджає користувача про потенційну загрозу.

У груповому чаті: Бот перевіряє посилання на наявність шкідливого контенту. Якщо посилання небезпечне, бот видаляє оригінальне повідомлення, дублює посилання, що можна побачити на рис. 3.12, форматуючи його під спойлер, та пише який користувач з посиланням на нього, намагався надіслати його та надсилає попередження.

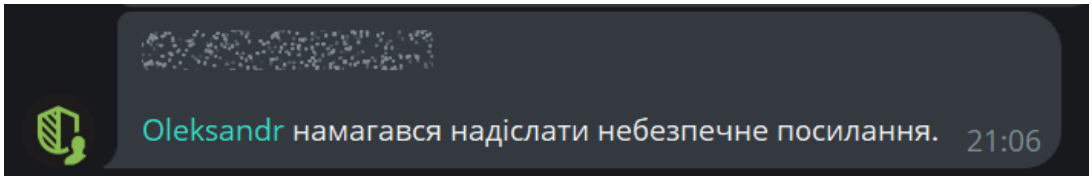

Рисунок 3.12 – Сповіщення від бота LinkCheck про спробу користувача надіслати небезпечне посилання.

Бот попереджає учасників групи та адміністраторів про потенційну загрозу.

Якщо ж повідомленні користувача був ще текст, для того аби сенс повідомлення не загубився, якщо користувач надіслав шкідливе посилання випадково, бот дублює його, це відображено на рис. 3.13.

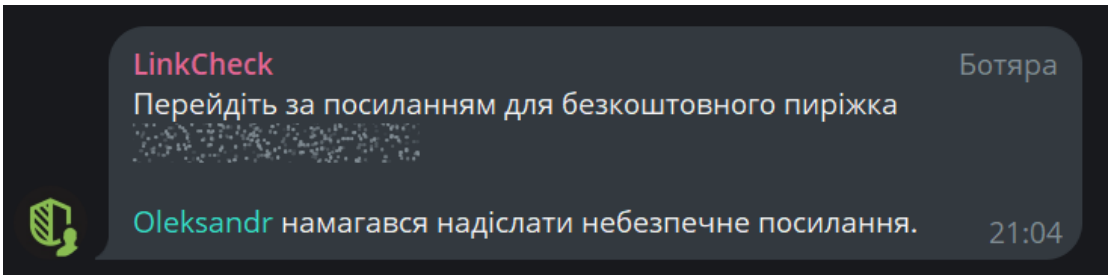

Рисунок 3.13 – Повідомлення від бота LinkCheck про спробу користувача надіслати потенційно небезпечне посилання якщо в повідомленні крім посилання був текст.

Бот інформує про загрозу та відображає текст повідомлення користувача.

Таким чином, бот забезпечує зручний інтерфейс для взаємодії з користувачами як у приватних чатах, так і у групових. Команди, такі як /start, /help, /support_dev, /language, дозволяють користувачам налаштовувати взаємодію з ботом та отримувати необхідну інформацію. Автоматична перевірка посилань допомагає забезпечити безпеку чатів.

3.3. Аналіз результатів тестування

Тестування проводилося в декілька етапів:

а) Функціональне тестування: Перевірка роботи основних команд бота, таких як /start, /help, /support_dev, /language. Команди тестувалися як у приватних чатах, так і в групових.

б) Тестування обробки посилань: Надсилання повідомлень з різними URL-адресами для перевірки їхньої безпеки. Випробовувалися як безпечні, так і небезпечні посилання.

в) Тестування мовних налаштувань: Перевірка коректності збереження та використання мовних налаштувань для користувачів і груп. Зміна мови взаємодії та перевірка відповідності текстів на обраній мові.

Функціональне тестування показало, що всі основні команди бота працюють коректно. У приватних чатах команда /start відображає привітання та пропонує додати бота до групи або обрати мову. У групових чатах команда /start, яка надсилається автоматично, після додавання бота у групу, відображає повідомлення

про успішне додавання бота та пропонує обрати мову. Команда /help правильно виводить список доступних команд у обох контекстах. Команда /support_dev надає інформацію про підтримку проекту. Команда /language дозволяє обрати мову з доступного списку.

Тестування обробки посилань продемонструвало, що бот успішно виявляє та перевіряє URL-адреси на наявність шкідливого контенту. У приватних чатах бот правильно інформує користувачів про безпеку посилання. У групових чатах бот видаляє небезпечні посилання та надсилає попередження. Перевірка посилань за допомогою API VirusTotal відбувається швидко та надійно.

Тестування мовних налаштувань підтвердило, що бот коректно зберігає та використовує мовні налаштування. Користувачі можуть змінювати мову взаємодії, і всі відповіді та повідомлення бота відображаються на обраній мові. Зміна мови у групових чатах працює лише для адміністраторів, що відповідає очікуванням.

ВИСНОВКИ

У процесі написання дипломної роботи було розроблено Telegram бот для перевірки безпеки посилань, який успішно вирішує завдання автоматичної перевірки URL-адрес на наявність шкідливого контенту. Бот був створений з використанням мови програмування Python та бібліотеки TeleBot (PyTelegramBotAPI), що дозволило забезпечити його функціональність, надійність та зручність у використанні.

Ініціалізація та конфігурація бота включали налаштування середовища розробки, отримання необхідних ключів доступу до Telegram API та VirusTotal API, а також налаштування конфігураційних файлів. Це забезпечило правильну роботу бота та його взаємодію з користувачами.

Реалізовані команди бота, такі як /start, /help, /support_dev, /language, а також обробка повідомлень з посиланнями, були протестовані в різних контекстах (приватні та групові чати). Бот коректно реагував на запити користувачів, забезпечуючи належний рівень функціональності.

Інтеграція з VirusTotal API дозволила боту ефективно перевіряти URL-адреси на наявність шкідливого контенту. Це забезпечило швидку і точну перевірку посилань та інформування користувачів про їхню безпеку.

Функціонал збереження мовних налаштувань користувачів був реалізований за допомогою JSON-файлу. Це дозволило боту підтримувати багатомовний інтерфейс і забезпечити зручність користувачів, що позитивно вплинуло на загальний користувацький досвід.

Проведене тестування підтвердило функціональність та надійність бота. Він успішно обробляв команди, перевіряв посилання та використовував мовні налаштування відповідно до очікувань, забезпечуючи високий рівень захисту та зручності для користувачів.

Завдяки використанню сучасних технологій і бібліотек, було створено ефективний інструмент для забезпечення безпеки користувачів у Telegram чатах. Бот автоматично перевіряє посилання, що надсилаються користувачами, і повідомляє про їхню безпеку, що значно підвищує захист від шкідливого контенту.

Крім того, багатомовна підтримка дозволяє користувачам взаємодіяти з ботом на зручній для них мові, що робить його ще більш доступним та корисним.

Загалом, розробка Telegram бота для перевірки безпеки посилань продемонструвала можливості автоматизації процесів захисту інформації в сучасних комунікаційних платформах. Результати роботи можуть бути використані для подальшого вдосконалення та розширення функціоналу бота, а також для розробки нових інструментів у сфері інформаційної безпеки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Python download. URL: https://www.python.org/downloads/ (Дата звернення: 19.02.2024)
2. Telegram bot library download. URL: https://pypi.org/project/python-telegram-bot/ (Дата звернення: 19.02.2024)
3. Python Official Documentation. URL: https://docs.python.org/3/ (Дата звернення: 04.04.2024)
4. Official PyCharm Community Edition. URL: https://www.jetbrains.com/pycharm/download/?section=windows (Дата звернення: 04.04.2024)
5. VirusTotal Official Documentation. URL: https://docs.virustotal.com/reference/overview (Дата звернення: 05.04.2024)
6. Official documentation of Telegram Bot API. URL: https://core.telegram.org/bots/api (Дата звернення: 04.04.2024)
7. Official documentation of the TeleBot (PyTelegramBotAPI) library. URL: https://github.com/eternnoir/pyTelegramBotAPI (Дата звернення: 04.04.2024)
8. Working with JSON Data in Python. URL: https://realpython.com/python-json/ (Дата звернення: 10.05.2024)
9. Стаття «The importance of API integration for modern business» URL: https://www.primenearshore.com/news-articles/the-importance-of-api-integration-for-modern-business (Дата звернення: 21.04.2024)
10. Vladyslav Zinchenko «Data Integration Using API: Essential Strategies and Benefits» - March 18, 2024
11. Tetiana Stoyko «What Is API Integration And How Does It Work?» - April 05, 2022
12. Automating-VirusTotal-APIv3-for-IPs-and-URLs. URL: https://github.com/b-fullam/Automating-VirusTotal-APIv3-for-IPs-and-URLs (Дата звернення 05.04.2024)

13. Official PyCharm documentation. URL: https://www.jetbrains.com/help/pycharm/ getting-started.html (Дата звернення: 04.04.2024)

ДОДАТОК

КОД TELEGRAM БОТА ДЛЯ ПЕРЕВІРКИ БЕЗПЕКИ ПОСИЛАНЬ

Вміст файлу config.txt

Для безпеки свого боту я не буду писати повний API ключ бота та ключ VirusTotal

TELEGRAM_BOT_TOKEN=71823**********blRCiFI

VIRUSTOTAL_API_KEY=51bb**********de24539d

ДОДАТОК А
КОД ОСНОВНОГО ФАЙЛУ MAIN.PY

```
import re
import requests
import telebot
from telebot import types
from LANGUAGE_FILE import load_language_preferences, save_language_preferences
from translations import translations

# Функція для зчитування конфігураційного файлу
def read_config(file_path):
    config = {}
    with open(file_path, 'r') as file:
        for line in file:
            key, value = line.strip().split('=')
            config[key] = value
    return config

# Зчитуємо конфігураційні параметри з файлу
config = read_config('config.txt')

# Ініціалізуємо бота з токеном з конфігураційного файлу
bot = telebot.TeleBot(config['TELEGRAM_BOT_TOKEN'])
VT_API_KEY = config['VIRUSTOTAL_API_KEY']  # Ключ для доступу до API VirusTotal
default_language = "en"  # Мова за замовчуванням
processed_messages = set()  # Набір оброблених повідомлень

# Функція для отримання перекладу
```

```
def get_translation(lang, key):
    return translations.get(lang, translations[default_language]).get(key, key)

# Завантаження мовних налаштувань користувачів і груп
language_preferences = load_language_preferences()
user_languages = language_preferences.get("user_languages", {})
group_languages = language_preferences.get("group_languages", {})

# Встановлення мови для користувача
def set_user_language(user_id, lang):
    user_languages[str(user_id)] = lang
    save_language_preferences({"user_languages": user_languages, "group_languages":
group_languages})

# Встановлення мови для групи
def set_group_language(chat_id, lang):
    group_languages[str(chat_id)] = lang
    save_language_preferences({"user_languages": user_languages, "group_languages":
group_languages})

# Отримання мови користувача
def get_user_language(user_id):
    return user_languages.get(str(user_id), default_language)

# Отримання мови групи
def get_group_language(chat_id):
    return group_languages.get(str(chat_id), default_language)

# Перевірка URL через VirusTotal
def check_url_vt(url):
```

```
    headers = {"x-apikey": VT_API_KEY}
    params = {"url": url}
    response = requests.post("https://www.virustotal.com/api/v3/urls", headers=headers, data=params)
    if response.status_code == 200:
        json_response = response.json()
        analysis_id = json_response["data"]["id"]
        analysis_url = f"https://www.virustotal.com/api/v3/analyses/{analysis_id}"
        analysis_response = requests.get(analysis_url, headers=headers)
        if analysis_response.status_code == 200:
            analysis_results = analysis_response.json()
            stats = analysis_results["data"]["attributes"]["stats"]
            if stats["malicious"] > 0:
                return True
    return False

# Обробник команди /start
@bot.message_handler(commands=["start"])
def main(message):
    if message.chat.type == "private":
        user_id = message.from_user.id
        lang = get_user_language(user_id)
        markup = types.InlineKeyboardMarkup()
        button_add = types.InlineKeyboardButton(
            text=get_translation(lang, "add_to_chat"), url="https://t.me/L1nk_Check_Bot?startgroup=new"
        )
        button_language = types.InlineKeyboardButton(
            text=get_translation(lang, "choose_language"), callback_data="choose_language"
```

```
        )
        markup.add(button_add)
        markup.add(button_language)
        bot.send_message(
            message.chat.id,
            get_translation(lang, "start_private_combined"),
            reply_markup=markup,
        )
    elif message.chat.type in ["group", "supergroup"]:
        chat_id = message.chat.id
        lang = get_group_language(chat_id)
        markup = types.InlineKeyboardMarkup()
        button_language = types.InlineKeyboardButton(
            text=get_translation(lang, "choose_language"),
callback_data="choose_language"
        )
        markup.add(button_language)
        bot.send_message(
            chat_id,
            get_translation(lang, "group_added_message"),
            reply_markup=markup
        )

# Обробник команди /help
@bot.message_handler(commands=["help"])
def help_command(message):
    if message.chat.type == "private":
        user_id = message.from_user.id
        lang = get_user_language(user_id)
    else:
```

```
        chat_id = message.chat.id
        lang = get_group_language(chat_id)
    bot.send_message(
        message.chat.id,
        get_translation(lang, "help")
    )

# Обробник команди /support_dev
@bot.message_handler(commands=["support_dev"])
def support_dev(message):
    if message.chat.type == "private":
        user_id = message.from_user.id
        lang = get_user_language(user_id)
    else:
        chat_id = message.chat.id
        lang = get_group_language(chat_id)
    bot.send_message(
        message.chat.id,
        get_translation(lang, "support_dev"),
        parse_mode="Markdown"
    )

# Обробник команди /language
@bot.message_handler(commands=["language"])
def language_command(message):
    if message.chat.type == "private":
        user_id = message.from_user.id
        lang = get_user_language(user_id)
    else:
        chat_id = message.chat.id
```

```
        lang = get_group_language(chat_id)
    markup = types.InlineKeyboardMarkup()
    for code, name in [("en", "English"), ("uk", "Українська"), ("ja", "日本語"), ("es",
"Español"), ("zh", "中文")]:
        markup.add(types.InlineKeyboardButton(text=name,
callback_data=f"set_language_{code}"))
    bot.send_message(
        message.chat.id,
        get_translation(lang, "choose_language"),
        reply_markup=markup
    )

# Обробник змін мови через callback
@bot.callback_query_handler(func=lambda call: call.data.startswith("set_language_"))
def handle_language_change(call):
    lang_code = call.data.split("_")[-1]
    if call.message.chat.type == "private":
        user_id = call.from_user.id
        set_user_language(user_id, lang_code)
        bot.answer_callback_query(call.id)
        bot.send_message(
            call.message.chat.id,
            get_translation(lang_code, "language_changed").format(lang_code)
        )
    else:
        chat_id = call.message.chat.id
        member = bot.get_chat_member(chat_id, call.from_user.id)
        if member.status in ['administrator', 'creator']:
            set_group_language(chat_id, lang_code)
```

```
            bot.answer_callback_query(call.id)
            bot.send_message(
                call.message.chat.id,
                get_translation(lang_code, "language_changed").format(lang_code)
            )
        else:
            current_lang = get_group_language(chat_id)
            bot.answer_callback_query(call.id,                text=get_translation(current_lang,
"not_authorized"), show_alert=True)

# Обробник вибору мови через callback
@bot.callback_query_handler(func=lambda call: call.data == "choose_language")
def choose_language(call):
    if call.message.chat.type == "private":
        user_id = call.from_user.id
        lang = get_user_language(user_id)
    else:
        chat_id = call.message.chat.id
        lang = get_group_language(chat_id)
    markup = types.InlineKeyboardMarkup()
    for code, name in [("en", "English"), ("uk", "Українська"), ("ja", "日本語"), ("es",
"Español"), ("zh", "中文")]:
        markup.add(types.InlineKeyboardButton(text=name,
callback_data=f"set_language_{code}"))
    bot.send_message(
        call.message.chat.id,
        get_translation(lang, "choose_language"),
        reply_markup=markup
    )
```

```
# Обробник повідомлень з посиланнями
@bot.message_handler(regexp=r"https?://[^\s]+")
def handle_links(message):
    if message.chat.type == "private":
        user_id = message.from_user.id
        lang = get_user_language(user_id)
        urls = re.findall(r"https?://[^\s]+", message.text)
        for url in urls:
            if check_url_vt(url):
                bot.reply_to(message, get_translation(lang, "unsafe_link"))
            else:
                bot.reply_to(message, get_translation(lang, "safe_link"))
    else:
        chat_id = message.chat.id
        lang = get_group_language(chat_id)
        if message.message_id in processed_messages:
            return
        processed_messages.add(message.message_id)
        urls = re.findall(r"https?://[^\s]+", message.text)
        dangerous_urls = []
        modified_text = message.text
        for url in urls:
            if check_url_vt(url):
                dangerous_urls.append(url)
                modified_text = message.text.replace(url, f"<tg-spoiler>{url}</tg-spoiler>")
        if dangerous_urls:
            try:
                bot.delete_message(message.chat.id, message.message_id)
            except:
```

```
            pass  # Ігноруємо помилки видалення
        user_link = f'<a href="tg://user?id={message.from_user.id}">{message.from_user.first_name}</a>'
        if len(dangerous_urls) == 1:
            warning_message = (
                f"{modified_text.strip()}\n\n"
                f"{get_translation(lang, 'dangerous_link_warning').format(user_link)}"
            )
        else:
            warning_message = (
                f"{modified_text.strip()}\n\n"
                f"{get_translation(lang, 'dangerous_links_warning').format(user_link)}"
            )
        bot.send_message(
            message.chat.id,
            warning_message,
            parse_mode="HTML",
            disable_web_page_preview=True,
        )

# Запуск бота у нескінченному циклі
bot.polling(none_stop=True)
```

ДОДАТОК Б

ВМІСТ ФАЙЛУ З ПЕРЕКЛАДАМИ TRANSLATIONS.PY

```
translations = {
    "en": {
        "start_private_combined": "Hello! I will monitor the safety of links sent in your chat.\nYou can also send me links in private messages to check their safety.",
        "start_group": "Hello! I am already monitoring the safety of links sent in your chat. Enjoy safe chatting!",
        "help": "Hello! I am LinkCheck, and I will monitor the safety of links sent in your chat.\n\n"
                "Commands:\n"
                "/start - In private messages, add the bot to a group / In a group, check if the bot is added\n"
                "/help - List of commands\n"
                "/support_dev - Support the developers\n"
                "/language - Change the bot language",
        "support_dev": "If you want to support the development of the project and our developers, please, you can make a donation\n"
                       "Monobank `5375411513195599`",
        "dangerous_link_warning": "{} tried to send a dangerous link.",
        "dangerous_links_warning": "{} tried to send dangerous links.",
        "choose_language": "Please choose your language:",
        "language_changed": "Language changed to {}.",
        "add_to_chat": "Add to chat",
        "not_authorized": "You do not have administrator rights to change the language.",
        "group_added_message": "Thank you for adding me in your group chat. I will check all links for safety.",
        "send_links_for_checking": "You can send me links in private messages to check their safety.",
```

```
        "safe_link": "This link is safe.",
        "unsafe_link": "This link is not safe."
    },
    "uk": {
        "start_private_combined": "Привіт! Я буду стежити за безпекою посилань, які надсилаються в твоєму чаті.\nТакож ви можете надсилати мені посилання в приватних повідомленнях для перевірки їх безпеки.",
        "start_group": "Привіт! Я вже стежу за безпекою посилань, які надсилаються в твоєму чаті. Насолоджуйтесь безпечною перепискою!",
        "help": "Привіт! Я LinkCheck, я буду стежити за безпекою посилань, які надсилаються в твоєму чаті.\n\n"
                "Список команд:\n"
                "/start - В приватних повідомленнях додати бота в групу / В групі перевірити чи додався бот\n"
                "/help - Список команд\n"
                "/support_dev - Підтримати розробників\n"
                "/language - Змінити мову бота",
        "support_dev": "Якщо ви хочете підтримати розвиток проекту та наших розробників, будь ласка, можете зробити внесок\n"
                       "Монобанк `5375411513195599`",
        "dangerous_link_warning": "{} намагався надіслати небезпечне посилання.",
        "dangerous_links_warning": "{} намагався надіслати небезпечні посилання.",
        "choose_language": "Будь ласка, оберіть вашу мову:",
        "language_changed": "Мова змінена на {}.",
        "add_to_chat": "Додати до чату",
        "not_authorized": "Ви не маєте прав адміністратора для зміни мови.",
        "group_added_message": "Дякую за додавання мене в вашу групу. Я буду перевіряти всі посилання на безпеку.",
```

"send_links_for_checking": "Ви можете надсилати мені посилання в приватних повідомленнях для перевірки їх безпеки.",

"safe_link": "Це посилання безпечне.",

"unsafe_link": "Це посилання небезпечне."

},

"ja": {

"start_private_combined": "こんにちは！チャットで送信されたリンクの安全性を監視します。\nまた、プライベートメッセージでリンクを送信して、安全性を確認することもできます。",

"start_group": "こんにちは！チャットで送信されたリンクの安全性をすでに監視しています。安全なチャットを楽しんでください！",

"help": "こんにちは！私はLinkCheckです。チャットで送信されたリンクの安全性を監視します。\n\n"

"コマンドリスト:\n"

"/start - プライベートメッセージでボットをグループに追加 / グループでボットが追加されたか確認\n"

"/help - コマンドリスト\n"

"/support_dev - 開発者を支援する\n"

"/language - ボットの言語を変更する",

"support_dev": "プロジェクトと開発者を支援したい場合は、寄付をお願いします\n"

"Monobank `5375411513195599`",

"dangerous_link_warning": "{} は危険なリンクを送信しようとしました。",

"dangerous_links_warning": "{} は危険なリンクを送信しようとしました。",

"choose_language": "言語を選択してください:",

"language_changed": "言語が {} に変更されました。",

"add_to_chat": "チャットに追加",

"not_authorized": "言語を変更する管理者権限がありません。",

"group_added_message": "グループチャットに私を追加してくれてありがとう。すべてのリンクの安全性を確認します。",

"send_links_for_checking": "プライベートメッセージでリンクを送信して、安全性を確認することもできます。",

"safe_link": "このリンクは安全です。",

"unsafe_link": "このリンクは安全ではありません。"

},

"es": {

"start_private_combined": "¡Hola! Supervisaré la seguridad de los enlaces enviados en tu chat.\nTambién puedes enviarme enlaces en mensajes privados para verificar su seguridad.",

"start_group": "¡Hola! Ya estoy supervisando la seguridad de los enlaces enviados en tu chat. ¡Disfruta de un chat seguro!",

"help": "¡Hola! Soy LinkCheck y supervisaré la seguridad de los enlaces enviados en tu chat.\n\n"

"Lista de comandos:\n"

"/start - En mensajes privados, agrega el bot al grupo / En un grupo, verifica si el bot se ha agregado\n"

"/help - Lista de comandos\n"

"/support_dev - Apoyar a los desarrolladores\n"

"/language - Cambiar el idioma del bot",

"support_dev": "Si deseas apoyar el desarrollo del proyecto y a nuestros desarrolladores, por favor, puedes hacer una donación\n"

"Monobank `5375411513195599`",

"dangerous_link_warning": "{} intentó enviar un enlace peligroso.",

"dangerous_links_warning": "{} intentó enviar enlaces peligrosos.",

"choose_language": "Por favor, elige tu idioma:",

"language_changed": "El idioma ha cambiado a {}.",

"add_to_chat": "Añadir al chat",

"not_authorized": "No tienes derechos de administrador para cambiar el idioma.",

"group_added_message": "Gracias por agregarme a tu grupo. Revisaré todos los enlaces por seguridad.",

"send_links_for_checking": "Puedes enviarme enlaces en mensajes privados para verificar su seguridad.",

"safe_link": "Este enlace es seguro.",

"unsafe_link": "Este enlace no es seguro."

},

"zh": {

"start_private_combined": "**你好！我会**监控你聊天中发送的链接的安全性。\n**你也可以在私人消息中**发送链接给我检查其安全性。",

"start_group": "**你好！我已**经在监控你聊天中发送的链接的安全性。享受安全的聊天！",

"help": "**你好！我是**LinkCheck，**我会**监控你聊天中发送的链接的安全性。\n\n"

"**命令列表**:\n"

"/start - **在私人消息中将机器人添加到群**组 / **在群**组中检查机器人是否已添加\n"

"/help - **命令列表**\n"

"/support_dev - **支持开**发者\n"

"/language - **更改机器人的**语言",

```
            "support_dev": "如果您想支持项目开发和我们的开发者，请捐款\n"
                           "Monobank `5375411513195599`",
            "dangerous_link_warning": "{} 试图发送危险链接。",
            "dangerous_links_warning": "{} 试图发送危险链接。",
            "choose_language": "请选择您的语言：",
            "language_changed": "语言已更改为 {}。",
            "add_to_chat": "添加到聊天",
            "not_authorized": "您没有更改语言的管理员权限。",
            "group_added_message": "谢谢你把我加入你的群聊。我会检查所有链接的安全性。",
            "send_links_for_checking": "你可以在私人消息中发送链接给我检查其安全性。",
            "safe_link": "这个链接是安全的。",
            "unsafe_link": "这个链接不安全。"
        }
    }
```

ДОДАТОК В
КОД ФАЙЛУ LANGUAGE_FILE.PY ЯКИЙ МІСТИТЬ ФУНКЦІЇ ДЛЯ РОБОТИ З ФАЙЛОМ-БАЗОЮ ДАНИХ JSON

```
import json
import os

LANGUAGE_FILE = 'language_preferences.json'  # Шлях до файлу з мовними налаштуваннями

# Ініціалізація мовних налаштувань
def initialize_preferences():
    if not os.path.exists(LANGUAGE_FILE):
        initial_preferences = {
            "user_languages": {},
            "group_languages": {}
        }
        with open(LANGUAGE_FILE, 'w') as file:
            json.dump(initial_preferences, file)

# Завантаження мовних налаштувань з файлу
def load_language_preferences():
    if os.path.exists(LANGUAGE_FILE):
        with open(LANGUAGE_FILE, 'r') as file:
            return json.load(file)
    return {"user_languages": {}, "group_languages": {}}

# Збереження мовних налаштувань до файлу
def save_language_preferences(prefs):
    with open(LANGUAGE_FILE, 'w') as file:
        json.dump(prefs, file, indent=4, ensure_ascii=False)
```

```
# Ініціалізація файлу, якщо його не існує
initialize_preferences()
```

ДОДАТОК Г

ВМІСТ ФАЙЛУ-БАЗИ ДАНИХ LANGUAGE_PREFERENCES.JSON ЯКИЙ ЗБЕРІГАЄ ОБРАНУ МОВУ КОРИСТУВАЧАМИ


```
{
   "user_languages": {
      "587096106": "uk"
   },
   "group_languages": {
      "-1001247417889": "uk",
      "-1002113040454": "ja"
   }
}
```